Description

Method and arrangement for predicting measurement data using given measurement data

The invention relates to a method and arrangement for predicting measurement data using given measurement data.

A technical system often requires facilities for forecasting based on known (measurement) data, particularly in the context of error susceptibility or cost estimates.

Forecasts generated by experts are generally subject to errors. Experts cannot carry out exact analyses, at least of highly complex systems.

A stochastic point process, in particular a Poisson process, is described in [1].

The **object** of the invention is to allow the automatic prediction (forecast) of measurement data using given measurement data.

This object is achieved in accordance with the features of the independent patent claims. Developments of the invention are described in the dependent claims.

In order to achieve the object, a method is proposed for predicting measurement data using given measurement data, in which a stochastic process is matched to the given measurement data. Simulation runs are carried out from a given time-point until a final time-point. The forecast measurement data is determined for each simulation run. Measurement data for the final time-point

is predicted within a range of values, which is governed by the forecast measurement data.

One development is to define a confidence range for the prediction of measurement data, where the a% lowest and b% highest forecast measurement data are eliminated. In particular, a% can equal b%. For example, a 95% confidence range can thus be defined by ignoring the 2.5% lowest and 2.5% highest forecast measurement data.

One advantage is that the measurement data can be predicted (forecast) with an accuracy that is within a confidence range, from a given time-point. This makes it possible to identify e.g. the feasibility or impossibility of a task associated with the measurement data, at an early stage. Appropriate measures can therefore be initiated in order to counteract forecast impossibility. This is particularly important in the case of a complex system, e.g. a software development process, where the extent to which a schedule can be followed before the software is completed can be shown in a subsequent test phase. Even more important in this context is the ability to adopt countermeasures at an early stage if a delay has been clearly identified, e.g. in an integration test phase. This firstly affects the feasibility of the specified deadline (timescale) and secondly directly affects costs, since non-compliance with the agreed timescale often results in additional costs.

One refinement is for the stochastic process to be a non-homogeneous Poisson process.

In particular, the measurement data may in one refinement comprise numbers of errors. This applies to software development,

for example, where the level of maturity is documented in accordance with the errors measured in a test phase. Completion is directly dependent on this level of maturity. In other words, the software cannot be delivered to customers until most of the errors have been removed from the software. This is particularly important with regard to resources (required to test and correct errors) and costs (due to delayed delivery).

In order to achieve the object, a method is also proposed for predicting measurement data using given measurement data, in which a stochastic process is matched to the given measurement data. A range is ascertained, by sorting the probability values generated by the stochastic process according to size, around an expected value. Measurement data is predicted on the basis of this range, and in particular the probability values within the range.

One development is for the probability values generated by the stochastic process to be sorted symmetrically by size around the expected value. In particular, this means that the highest probability value represents the middle of the range, i.e. the expected value, whereas the next highest probability value is arranged to the right or left of the expected value. The next highest probability value is then arranged symmetrically on the other side of the expected value, in turn.

This analytical (design) procedure provides a range, where the breadth of the range in turn indicates which probability values are significant in the prediction of the measurement data.

In one particular refinement, the breadth of the range is determined by ignoring the

probability values that lie below a given threshold.

This produces a range (confidence range), which has a specific breadth as a result of the threshold. This breadth corresponds to the certainty with which the measurement data is predicted.

If one assumes that the stochastic process is a non-homogeneous Poisson process, then the non-homogeneous Poisson process defines a step size, particularly on a time axis t, which indicates when the next error will occur. One characteristic of the non-homogeneous Poisson process is that it has no memory, so that a "no-memory" search is carried out from each error that occurs at a specific time-point, for a time-point that indicates the next error.

In order to achieve the object, an arrangement is also proposed for predicting measurement data using given measurement data, whereby a processor unit is provided and configured in such a way that:

a) a stochastic process can be matched to the given measurement data;
b) simulation runs can be carried out from a given time-point until a final time-point;
c) the forecast measurement data can be determined for each simulation run;
d) the prediction of measurement data for the final time-point can be predicted within a range of values, which is determined by the forecast measurement data.

In order to achieve the object, an arrangement is further proposed for predicting measurement data using given measurement data, whereby a processor unit is provided and configured in such a way that:

a) a stochastic process can be matched to the given measurement data;

b) a range can be ascertained by sorting probability values generated by the stochastic process according to size around an expected value;

c) the measurement data is predicted within the limits of the range.

The arrangements are particularly suitable for carrying out the inventive method or the developments described above.

Exemplary embodiments of the invention are shown and explained below with reference to the drawings, in which:

Fig. 1 is a graph showing an accumulated number of errors over a test period;

Fig. 2 is a graph showing the superimposed confidence ranges for different process models;

Fig. 3 is a flowchart showing the steps in a method for predicting measurement data using given measurement data;

Fig. 4 is a further flowchart showing the steps in a method for predicting measurement data using given measurement data;

Fig. 5 shows a processor unit.

In order to be able to forecast a number of errors to be expected in a technical process, e.g. in a software development process, non-homogeneous Poisson

processes (NHPP) are calibrated, i.e. matched to measurement data, e.g. the occurrence of errors over time, as follows:

The following equation describes a counting process associated with the stochastic point process (non-homogeneous Poisson process):

$$\{N(t)\}_{t \in \mathbf{R}^+} \tag{1}$$

and a time-point t_0 defines the end of a test period, i.e. a time-point at which the given data ends. The stochastic processes

$$\{U(t)\}_{t \in \mathbf{R}^+} \quad \text{and} \tag{2}$$

$$\{O(t)\}_{t \in \mathbf{R}^+} \tag{3}$$

are searched with

$$P\left(U(t) \leq N(t) - N(t_0) \leq O(t) \,\middle|\, N(t_0) = n_0\right) \geq \alpha \tag{4},$$

for all time-points where $t>t_0$ and given values $\alpha \in (0.1)$ (confidence level) and $n_0 \in \mathbf{N}$. In particular, the following text examines the increases in the stochastic countings process in relation to the time-point t_0.

In the present case, where equation (1) represents a non-homogeneous Poisson process, the following equation (cf. [1])

$$P\left(N(t_1) - N(t_0) = \ell\right) = \exp\left(-\left[i(t_1) - i(t_0)\right]\right) \cdot \frac{\left[i(t_1) - i(t_0)\right]^{\ell}}{\ell!} \tag{5}$$

applies for

$$0 \leq t_0 < t_1 < \infty, \quad \ell \in \mathbf{N}_0 \tag{6}$$

and an intensity (mean measure, mean value function) of

$$i: \mathbf{R}^+ \rightarrow \mathbf{R}^+, t \mapsto i(t) = EN(t) \qquad (7).$$

Since the nature of the Poisson process dictates that the increases (error increases in this case) are independent of previous increases, equation (5) for the time-points $t > t_0$ to define a (minimum) range

$$[g_u, g_o] \equiv [g_u(t), g_o(t)] \subset \mathbf{N}_0 \qquad (8)$$

can be simplified to

$$\sum_{\ell=g_u}^{g_o} P\left(N(t) - N(t_0) = \ell\right) \geq \alpha \qquad (9).$$

Due to the unimodal nature of the Poisson count density, a range $[g_u, g_o]$ can be determined as follows:

Step 1: Sort the elementary probabilities

$$p_\ell := P\left(N(t) - N(t_0) = \ell\right), \ \ell \in \mathbf{N}_0$$

into descending order and label the values sorted thus using

$p_{(0)}, p_{(1)}, \ldots$ (i.e. $\{p_0, p_1, \ldots\} = \{p_{(0)}, p_{(1)}, \ldots\}$ and $p_{(0)} \geq p_{(1)} \geq \ldots$);

Step 2: Determine $\ell_{min} := \min\left\{\ell \in \mathbf{N}_0 \,\middle|\, \sum_{i=0}^{\ell} p_{(i)} \geq \alpha\right\}$; ;

Step 3: Determine an index set

$$I := \left\{i_0, \ldots, i_{\ell_{min}}\right\} \subset \mathbf{N}_0 \quad \text{where}$$

$$\left\{P_{i_0}, \ldots, P_{i_{\ell_{min}}}\right\} = \left\{P(0), \ldots, P(\ell_{min})\right\};$$

Step 4: Substitute $g_u := \min_{i \in I}\{i\}$ and $g_o := \max_{i \in I}\{i\}$.

The range from equation (8) is also referred to as the forecast range.

Stochastic simulation (second approach)

It is possible to determine the confidence range described using simulation, with the following steps:

Step 1: Start independent simulation runs based on the selected process model at time-point t_0 of the last error message $m \in \mathbf{N}$;

Step 2: End a simulation run as soon as the required final time-point t_e is reached;

Step 3: Repeat Step 2 until all simulation runs are finished;

Step 4: Sort the numbers $\hat{N}_i(t_e)$ of the errors generated in the i-th simulation run in the time period (t_0, t_e), i=1, ..., m, in descending order, and label the values sorted thus $\hat{N}_{(1)}(t_e), \ldots, \hat{N}_{(m)}(t_e)$;

Step 5: Substitute

$$\hat{g}_u := \hat{N}_{(\lfloor m \cdot \alpha/2 \rfloor)}(t_e) \quad \text{and}$$

$$\hat{g}_o := \hat{N}_{(\lceil m \cdot (1-\alpha/2) \rceil)}(t_e),$$

i.e. eliminate the (100?(1-α)/2)% lowest and highest values.

This produces the confidence range directly.

Each individual simulation run is based on a simulation algorithm, which is known from (cf. [2]):
Simulated generation of intermediate arrival times for a non-homogeneous Poisson process:

Step 1: Substitute $\bar{\lambda} := \sup_{t \geq t_s} \{\lambda(t)\}$, where:

$$\lambda(t) := \left.\frac{di}{dt}\right|_t \qquad (10).$$

Step 2: Generate a (pseudo) random variable X that is exponentially distributed with the parameter $\bar{\lambda}$, i.e. $X := -\log(U) / \bar{\lambda}$, where U is equally distributed over (0,1).

Step 3: Generate a random variable U that is equally distributed over (0,1).

Step 4: If $U \leq \lambda(t_s + X) / \bar{\lambda}$, then substitute $t^* := t_s + X$; otherwise substitute $t_s := t_s + X$ and go to Step 1.

The example graph in **Fig. 1** shows an accumulated number of errors during a given test period. From time-point t_0, it shows a prediction range for all time-points $t_0 + x$.

The intensity i is normally derived from equation (10) for λ. For example the result is as follows:

a) $\lambda(t) = a \cdot b \cdot c \cdot \exp\left(-bt^c\right) \cdot t^{c-1}$

($\lambda(t)$ is strictly monotonously descending for $c \leq 1$, and unimodal for $c > 1$ with a definitive maximum at a point

$t_{max} = \sqrt[c]{\frac{c-1}{bc}}$).

b) Otherwise, $\bar{\lambda}$ is derived in accordance with the above comments as follows:

$$\bar{\lambda} = \begin{cases} \lambda(t_s), & (c \leq 1) \vee (t_s \geq t_{max}) \\ \lambda(t_{max}) \end{cases}$$

The graph in **Fig. 2** shows the superimposed confidence ranges. In particular, this illustrates that possible forecasts become more scattered the further they extend into the future. In particular, confidence ranges calculated using different process models can be demonstrated in the same way as shown in Fig. 2.

Fig. 3 shows a flowchart for the steps of a method for predicting measurement data using given measurement data. In Step 301, a stochastic process, in particular a non-homogenous Poisson process (to represent a stochastic count process), is matched to given measurement data. In Step 302, simulation runs are run from time-point t_0 to a final time-point t_e that is to be forecast. In Step 303, for each simulation run, forecast measurement data is determined and a prediction of measurement data is restricted to a range which is covered by the measurement data determined by the simulation runs (see Step 304). In Step 305, a confidence range is determined in which a given proportion of the lowest and highest forecast measurement data is ignored in each case (this corresponds to the aforementioned range). The method terminates in Step 306.

Fig. 4 shows a further flowchart for the steps of a method for predicting measurement data using given measurement data. In Step 401, a stochastic process, in particular a non-homogenous Poisson process, is matched to the given measurement data. Probability values are determined using the stochastic process, and these are sorted according to size around an expected value (see Step 402). This sort operation results in the definition of a range, namely a confidence range in this case. The breadth of the confidence range is determined by comparing the accumulated probabilities with a given threshold. As described above, the confidence range gives a distribution or uncertainty, respectively, of a time-point t_0 in the future, which allows the measurement data to be estimated in the future (see Step 403). The method terminates in Step 404.

Fig. 5 shows a processor unit PRZE. The processor unit PRZE comprises a processor CPU, a memory unit MEM, and an input/output interface IOS, which is used in different ways via an interface IFC: a graphics interface allows output to be viewed on a monitor MON and/or output to a printer PRT. Inputs are entered via a mouse MAS or a keyboard TAST. The processor unit PRZE also includes a data bus BUS, which provides the connection between a memory unit MEM, the processor CPU and the input/output interface IOS. It is also possible to connect additional components to the data bus BUS, e.g. additional memory, data storage (hard disk) or scanner.

The C programming language is used in the following examples, which show an algorithm to define confidence ranges for forecasts and an algorithm for simulated definition of confidence ranges for forecasts.

Program 1:

```
/* Definition of confidence ranges for forecasts */
/* based on the generalized Goel-Okomoto model */

  #include <stdlib.h>
  #include <math.h>
  #include <stdio.h>

  #define true  1
  #define false -1

  double mv_genGO(double,double,double,double); double poisson(double,long); void ki_nhpp();
  int main(argc,argv)
  int argc;
  char *argv[];
  {
  double a,b,c,bt,st,kn;
  long low,upp,lauf;

  if (argc<7) {
  printf("\n\nZuwenig Argumente! \n\n");
  printf("Aufruf: %s <Par1> <Par2> <Par3> <Startzeit> <Endzeit>",
  "<KNiveau>\n\n", argv[0]); return 1;
   }

  a = atof(argv[1]);
  b = atof(argv[2]);
  c = atof(argv[3]);
  bt= atof(argv[4]);
  st= atof(argv[5]);
  kn= atof(argv[6]);

  for (lauf=1;lauf< ;lauf++) {
  ki_nhpp(mv_genGO,a,b,c,bt,bt+lauf*(st-bt)/10.,kn,&low,&upp);
  printf("Zeitpunkt: %8.2f    Fehlerintervall: [%d,%d]\n",
  bt+lauf*(st-bt)/10., low, upp);
   }
  return 0;
  }

  double mv_genGO(x,a,b,c)
  double x,a,b,c;
  { return( a*(1.0-exp(-b*pow(x,c))) ); }

  double poisson(lambda,wert)
  double lambda;
  long wert;
  {
  long i;
  double itval,hv;

  if (lambda<600) {
  itval = exp(-lambda);
  for (i=wert;i>=1;i--) { itval *= lambda/(double)i; }
   }
```

```
else {
hv = exp(-lambda/(double)wert);
itval = 1.0;
for (i=wert;i>=1;i--) { itval *= lambda/(double)i*hv; }
 }
return ( itval );
}

void ki_nhpp(mv_nhpp, par1_nhpp, par2_nhpp, par3_nhpp,
start_time, stop_time, k_niveau, lower, upper) double mv_nhpp(double,double,double,double);
   double par1_nhpp, par2_nhpp, par3_nhpp, start_time, stop_time, k_niveau; long *lower, *upper;
{
long lauf;
int lborder,mod_low,mod_upp;
double sum,tmp_mv, val_l, val_u;

tmp_mv = mv_nhpp(stop_time,par1_nhpp,par2_nhpp,par3_nhpp) -
mv_nhpp(start_time,par1_nhpp,par2_nhpp,par3_nhpp); lauf = (long)tmp_mv;
*lower = lauf-1;
*upper = lauf+1; mod_low= false; mod_upp= false; sum   = poisson(tmp_mv,lauf); val_l =
   poisson(tmp_mv,*lower); val_u  = poisson(tmp_mv,*upper);
while (sum<k_niveau) {
if (val_l<val_u) {
sum += val_u;
(*upper)++;
lborder = false;
mod_upp = true;
val_u  = poisson(tmp_mv,*upper);
  }
else {
sum += val_l;
(*lower)--;
lborder = true;
mod_low = true;
val_l  = poisson(tmp_mv,*lower);
  }
 }

if (lborder == true) { (*lower)++; }
else                 { (*upper)--; }

if (mod_low == false) { (*lower)++; }
if (mod_upp == false) { (*upper)--; }

return;
}
```

Program 2:

```
/*  Simulated definition of confidence ranges for
forecasts */
/* based on the generalized Goel-Okomoto model */

  #include <stdlib.h>
  #include <math.h>
  #include <time.h>
  #include <stdio.h>
  #include <values.h>

  #define true  1
  #define false -1

  double drand48(void);
  void srand48(long);

  double sim_exp(double); double lambda_genGO(double,double,double,double); void sim_nhpp();
  int main(argc,argv)
  int argc;
  char *argv[];
  {
  time_t t; double a,b,c,bt,st,pnt[1000000],check_time[12]; long lauf,no_pnt,seed_run; int clauf;
  FILE *datei;
  if (argc<6) {
  printf("\n\nZuwenig Argumente! \n\n");
  printf("Aufruf: %s <Par1> <Par2> <Par3> <Startzeit> <Endzeit>\n\n",
  argv[0]); return 1;
   }

  datei = fopen("sim.seed","r");
  if (datei==NULL) {
  seed_run = 1;
   }
  else {
  fscanf(datei,"%6d",&seed_run); fclose(datei); seed_run++;
   }

  datei = fopen("sim.seed","w+");
  fprintf(datei, "%6d\n", seed_run );
  fclose(datei);

   time (&t) ;                /* Initialisierung des     */
   t += seed_run*100 ;        /* Zufallszahlengenerators */
  srand48 ((unsigned long) t) ;  /* mit Hilfe der Systemzeit */
  a = atof(argv[1]);
  b = atof(argv[2]);
  c = atof(argv[3]);
  bt= atof(argv[4]);
  st= atof(argv[5]);

  sim_nhpp(lambda_genGO,a,b,c,bt,st,&pnt,&no_pnt);
  for (lauf=1;lauf<=no_pnt;lauf++) {
  printf("%15.7f %10d \n", pnt[lauf], lauf);
   }
```

```
datei = fopen("ki.tmp","a");
for (lauf=1;lauf< ;lauf++) {
check_time[lauf] = bt+lauf*(st-bt)/10.;
 }
check_time[11] = pnt[no_pnt]+1; /* größer als die größte
simulierte Zeit */
clauf = 1;
for (lauf=1;lauf<=no_pnt;lauf++) {
while (pnt[lauf]>=check_time[clauf]) { fprintf(datei, "%8.2f %6d ", check_time[clauf], lauf-1); clauf++;
  }
 }

if (pnt[no_pnt] <check_time[10]) {
for (lauf=clauf;lauf< ;lauf++) {  fprintf(datei, "%8.2f %6d ", check_time[lauf], no_pnt);
  }
 }

fprintf(datei, "\n");
fclose(datei);

return 0;
}

double sim_exp(lambda)
double lambda;
{ return( -log(drand48())/lambda ); }

double lambda_genGO(x,a,b,c)
double x,a,b,c;
{ return( a*b*c*pow(x,c-1)*exp(-b*pow(x,c)) ); }

void sim_nhpp(lambda_nhpp, par1_nhpp, par2_nhpp, par3_nhpp,
start_time, stop_time, path, no_points) double lambda_nhpp(double,double,double,double); double
    par1_nhpp, par2_nhpp, par3_nhpp, start_time, stop_time; double path[]; long *no_points;
{
double sim_time,x,u,x_bar,lambda_bar;
*no_points=0;
sim_time = start_time;

do {
if (par3_nhpp<=1) { lambda_bar = lambda_nhpp(sim_time,par1_nhpp,par2_nhpp,par3_nhpp);
  }
else {
x_bar = pow((par3_nhpp-1.0)/par2_nhpp/par3_nhpp,1.0/par3_nhpp);
if (sim_time>=x_bar) {
lambda_bar = lambda_nhpp(sim_time,par1_nhpp,par2_nhpp,par3_nhpp);
   }
else {
lambda_bar = lambda_nhpp(x_bar,par1_nhpp,par2_nhpp,par3_nhpp);
   }
  }

x = sim_exp(lambda_bar);
u = drand48();
```

```
if (u<=lambda_nhpp(sim_time+x,par1_nhpp,par2_nhpp,par3_nhpp)/lambda_bar) { (*no_points)++;
    path[*no_points]=sim_time+x;
   }
sim_time+=x;
 }
while (sim_time<=stop_time);
return;
}
```

Program 3:

```
/* Definition of confidence ranges from the simulation
data */
/* (the simulation data is sorted into ascending order)
*/
  #include <stdlib.h>
  #include <math.h>
  #include <stdio.h>

  int qsort_icmp(int*,int*);
  int qsort_icmp(x,y)
  int *x, *y;
  {
  if (*x<*y)      { return ( -1 ); }
  else if (*x==*y) { return (  0 ); }
  else            { return (  1 ); }
  }

  int main(argc,argv)
  int argc;
  char *argv[];
  {
  int pnt[11][100000];
  int qs[100000];
  char *dname;
  int frac,i;
  long lauf,lower_bound,upper_bound;
  long l,no_pnt,seed_run;
  double ctime[11],x;
  FILE *datei;


  if (argc<3) {
  printf("\n\nZuwenig Argumente! \n\n"); printf("Aufruf: %s <Dateiname> <Konfidenzniveau (in
     %%)>\n\n", argv[0]); return 1;
   }

  dname = argv[1];
  frac  = 100-atoi(argv[2]);
  lauf = 0;

  datei = fopen(dname,"r");
  if (datei==NULL) { return 1; }
  else {
  while (!feof(datei)) {
  lauf++;
  for (i=1;i<=9;i++) {
  fscanf(datei,"%8lf %6d ", &ctime[i],  &pnt[i][lauf]);
      }
  fscanf(datei,"%8lf %6d \n", &ctime[10],  &pnt[10][lauf]);
     }
  fclose(datei);
   }

  lower_bound = (long)floor(lauf*frac/200.); upper_bound = (long)ceil(lauf*(200.-frac)/200.);
  if (lower_bound<1) {lower_bound = 1;}
```

```
printf("\n\n%2d%%-Sicherheitsbereich bei %d Simulationslaufen\n\n",
100-frac,lauf);
for (i=1;i< ;i++) {
for (l=1;l<=lauf;l++) {
qs[l] = pnt[i][l];
   }

qsort(&qs[1], lauf, sizeof(int), &qsort_icmp);
printf("Zeitpunkt: %8.2f    Fehlerintervall: [%d,%d]\n",
ctime[i], qs[lower_bound], qs[upper_bound]);
 }

return 0 ;
}
```

References:

[1] Sidney I. Resnick: "Adventures in Stochastic Processes", Birkhäuser Boston, 1992, ISBN 3-7643-3591-2, pp. 303-317

[2] Brately et al., 1987